SIMULATION, ANIMATION, AND CONTROL OF MULTI–BODY MECHANISMS

John E. Hogan
Georgia Institute of Technology
School of Mechanical Engineering
Atlanta, GA 30332
(404) 894–8165

Wayne J. Book
Georgia Institute of Technology
School of Mechanical Engineering
Atlanta, GA 30332
(404) 894–3247

ABSTRACT

A three–dimensional graphical environment, called the Multi–Body Simulator or MBSim, has been developed for the simulation, animation, and control of multi–body mechanisms with or without sensors. The mechanisms can be composed of holonomic links, such as revolute and prismatic links, and nonholonomic links, such as tricycle like vehicles. This graphical environment is applied to tasks in the nuclear industry. The graphical environment has been designed and programmed in an object oriented fashion to enable a building block approach to creating multi–body mechanism models. The uses in the nuclear industry, attributes of this graphical environment, benefits of object oriented design, and aspects of simulation and animation with the system will be discussed.

I. MOTIVATION

The motivation for developing the MBSim system was for path planning and obstacle avoidance during the inspection of low level radioactive waste with autonomous mobile robots at the Savannah River Site. The simulation aspect of the system is used to develop and test path planning and obstacle avoidance schemes before implementation on actual robots. This includes the testing and analysis of sensor placement on the robots. The animation aspect of MBSim can be used to provide feedback from the actual devices to human operators. The control aspect can be used to direct and control the robot during the inspection of radioactive waste.

Two vehicles used at Savannah River, Stored Waste Autonomous Mobile Investigator (SWAMI) and Mobile Radio–controlled Teleoperator (MoRT), are modeled in Figures 1 and 2. SWAMI is used for inspecting barrels with stored radioactive waste. MoRT is used for mobile manipulation. Sensor placement on these vehicles is being analyzed

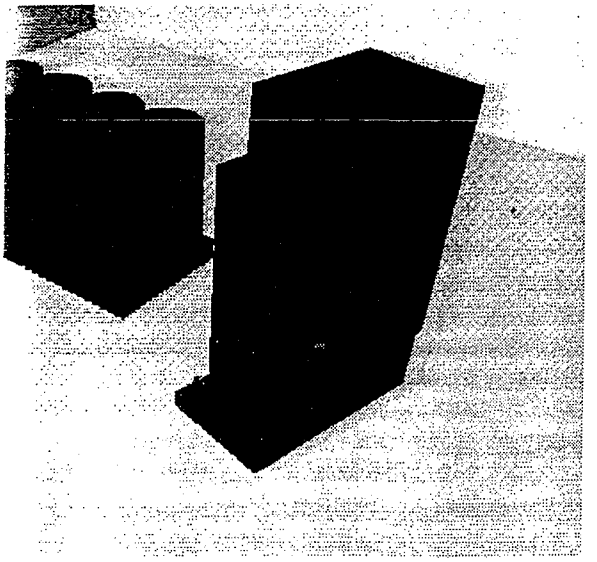

Figure 1 – SWAMI

Figure 2 – MoRT

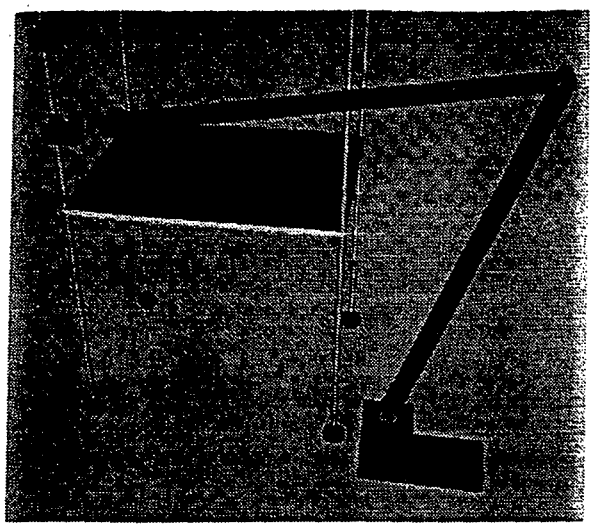

Figure 3 – RALF

through simulation using MBSim. The Robotic Arm Large and Flexible (RALF) developed at Georgia Tech is shown in Figure 3 (currently, RALF is modeled with simple rigid links). RALF is currently being developed for the removal of nuclear waste. Real–time joint position data will be fed from the teleoperated RALF into MBSim and displayed to the operator for feedback.

Commercial robotics packages typically do not enable the flexibility to easily and properly model kinematically constrained (nonholonomic) joints, include sensors, or enable real–time data to be fed in and displayed. Those that do may be expensive or require complex work arounds.

II. BASIC ELEMENTS

The basic elements of MBSim are mechanisms, environments, and views. The mechanisms are serial chains of links such as tricycle bases and revolute joints. Mechanisms can be stationary, such as an articulated industrial robot, or mobile, such as an automobile. The object oriented design allows the equations and functions specific to each link, such as kinematics and constraints, to be contained within each link object. This enables mechanisms to be easily developed with links in any combination and order. Table 1 shows the currently implemented link types, holonomic or nonholonomic, and example uses.

Each link can have any number of sensors attached, such as laser rangefinder and ultrasonic sensors. Since these sensors are attached to an individual link, they move with the link. The

Link Type	Example of Uses
Revolute (H)	articulated robots, manipulators
Prismatic (H)	XY robots, z–axis on SCARAs
Knife (N)	omni–directional mobile robots
Tricycle (N)	forklift like, car like robots
Dual Wheel (N)	two independently driven wheels, wheelchair and tank like robots

(H) indicates a holonomic link
(N) indicates a nonholonomic link

Table 1 – Link type and uses

graphics representing the link are constructed from graphic primitives, such as rectangular prisms and cylinders. The graphics can be as detailed or sparse as the user likes. There is a trade off between detail and speed. As the number of graphics primitives increase, the simulation performance decreases. The environment objects are stationary objects that are composed from the same graphics primitives used for link graphics. Together, mechanisms and environment objects allow users to build up most scenarios of interest. The view objects are windows in which the user sees the simulation. Each view enables the user to position the location of the view's eye and the point of interest. This enables virtually any view of the simulation to be seen and studied. Interactive zooming, translation, and rotation is under development.

III. OBJECT ORIENTED APPROACH

MBSim is designed in an object oriented fashion which enables the user to more easily construct many different scenarios to test concepts and algorithms [1]. Using the existing link types, the user defines a link by specifying the appropriate link parameters. For example, the tricycle requires the link parameters, length and distance between the rear wheels while the revolute link requires the Denavit–Hartenberg parameters which are defined in standard robotics texts [2].

MBSim utilizes script files to create the simulation. Script files are simple text files that the user edits to define each mechanism's description, link's graphics description, sensor description, and graphics view description. Thus, when using the existing link types, the user does not need to recompile for each different scenario or mechanism, but simply modifies the script files and re–runs the program. If an additional type of link is needed, the new link is developed by inheriting basic functions from the generic link object [3]. The

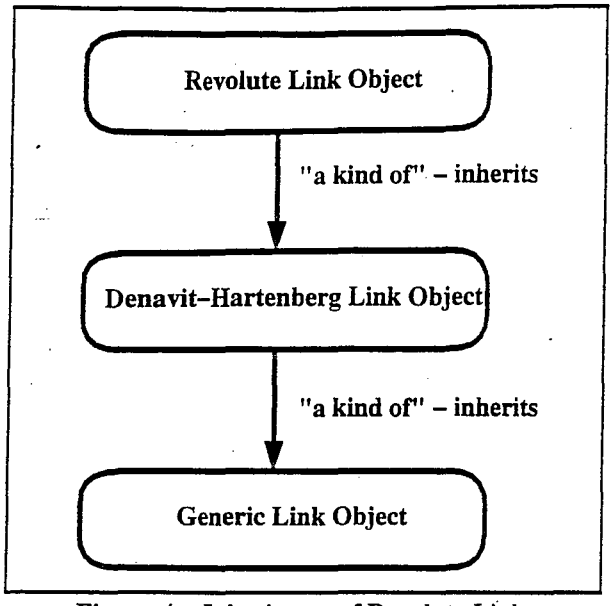

Figure 4 – Inheritance of Revolute Links

inheritance relationship can be found by using the phrase "a kind of" between two objects, such as "a revolute link is a kind of Denavit–Hartenberg link." Inheritance reduces the development effort required for new links and is a powerful feature of object oriented programming. The user need only define information and functions specific to the particular link since the basic functions are inherited from the generic link object. For example, both revolute and prismatic links can be defined with Denavit–Hartenberg (D–H) parameters. Information and functions common to all D–H links were defined in a D–H link object. Only information and functions specific to the revolute link was defined in the revolute link object. The basic D–H information and functions were inherited by the revolute link to form a usable link object. Figure 4 shows the inheritance of a revolute link object..

The link kinematics are a particularly useful example of the specific link functions. Each link contains its kinematic equations relative to the previous link and are accessed through common function names inherited from the generic link object. Thus, for each new combination of links that create a mechanism, the overall kinematic equations do not have to be developed. The mechanism object simply calls the common functions in each link object as needed to calculate the desired quantity. In a similar way, to integrate velocities to positions, each link knows how to integrate its velocity joint variables to obtain its position joint variables. Any constraint equations, such as in nonholonomic links, are applied here. The mechanism object which contains link objects calculates the dynamics using an iterative Newton–Euler method which calls the kinematic functions and mass properties in each link and propagates the velocities, accelerations, and forces from link to link. Again, no new derivation of dynamic equations for each different mechanism configuration.

The graphics are also objects which enable portability from machine to machine since all machine specific graphics calls are inside the graphics object and transparent to the user. We have converted from a Sun workstation utilizing the SPHIGS graphics library to a Silicon Graphics workstation using SGI's Graphics Language (GL) successfully.

IV. SIMULATION

The primary uses for MBSim utilizes the simulation aspect. Currently, the MBSim system is being used to develop path planning and obstacle avoidance schemes and determine sensor placement on autonomous robotic vehicles. The environments that the vehicles, SWAMI and MoRT, operate in are highly constrained and contain unknown obstacles. Due to the vehicles physical shape and kinematic properties, advanced motion schemes are required. The simulation performs motion using high level control schemes, such as giving velocity commands to the robot. This follows the higher level control found on many mobile robots. It is assumed the low level control will obtain the desired velocity quickly since the speeds we are concerned with are less than 5 inches per second. Low level control could be implemented in the future. As a check, the inverse dynamics can be calculated to ensure the torques and forces do not exceed the rated capacity of the motors.

The system can also be used to determine where to place them and how many sensors to use for the best performance. The analysis of sensor placement and number of sensors on these mobile robots will ensure adequate detection of the environment by the sensor arrays. Feedback from the sensors will be used to feed data into the control schemes to create autonomous motion.

Simulations with various configurations of sensors and environments can be run and compared easily. Existing simulation packages that perform similar functions are expensive and typically do not

include kinematically constrained links and sensors. Also, the source code for existing packages is generally not available, thus future additions and enhancements are difficult.

V. ANIMATION

The animation aspect is used with actual robots to observe their location and detect potential problems using the three dimensional graphical display of MBSim. In many situations the operator can not see or be near the robot due to a hazardous or inaccessible environment. The operator needs some way to view the conditions of the robot. Video cameras on the robot require a high volume of data to be transferred back to the operator. High volumes of data transferred from untethered autonomous robots is undesirable. The video cameras are typically stationary with only pan, tilt, and possibly zoom capabilities. An alternative is to use the feedback from the actual robot such as positions and sensor readings to inform the operator the status of the robot using MBSim to display a real–time model of the environment. MBSim enables the operator to move his viewpoint anywhere to obtain the best vantage point for a given situation. The total number of video cameras can be significantly reduced or eliminated entirely. The volume of data compared to multiple cameras can therefore be significantly reduced with little information lost. Since the data is being returned from the robot, the history of the motions and sensor readings from the robot can be easily stored and used for future analysis.

We are currently implementing this aspect with the teleoperated RALF. The combination of video cameras and the MBSim display will enable the operator to perform the jobs more accurately and efficiently. We are using one overall video camera view of the workcell mounted away from the robot and one detail video camera view mounted near the tip of the robot along with sending joint and sensor data to MBSim to be displayed in real–time to the operator.

Another use of the animation aspect of MBSim is to feed data into a robot model from output created by another outside program. The user gets a better feel for the performance of the robot than from plots alone. The animation conveys the user's work in a clear and intuitive way to others.

VI. CONTROL

The control aspect is being studied using simulation and animation features of MBSim. The control schemes developed and tested would typically be implemented on the robot itself rather than on MBSim. If the communication speed and reliability were good enough, control from afar would be feasible and enable a larger and lower cost computer to be utilized. The size, weight, and expense of the robot would be reduced which could be helpful in many situations. More common would be to mainly utilize the animation aspect. If for some reason a problem occurs with the robot where it can not move properly and the control scheme fails under these conditions, the human operator could invoke manual teleoperation through MBSim to finish the job or move the robot to a location where it can be safely repaired.

VII. CONCLUSION

MBSim is a multi–use flexible three–dimensional graphical environment for the simulation, animation, and control of multi–body mechanisms. The object oriented design enables many different mechanisms to be simulated and studied easily without the need of recoding and recompiling for every new configuration. The multi–body simulator can be used in the nuclear industry for system design, algorithm development, and sensor placement.

ACKNOWLEDGMENTS

This research has been partially funded by the Educational Research Development Association (ERDA) of the Georgia Universities and by the Westinghouse Savannah River Company under EDRA contract number 92006.

REFERENCES

1. Schmitt, Hogan, Cameron, and Book, "Integrated Modeling, Simulation, and Animation or Rigid Arms and Vehicles through Object–Oriented Programming", *Proceedings of the IMACS Symposium on Mathematical Modelling*, MATHMOD '94, vol. 3, p. 566, (1994)

2. Craig, John J., *Introduction to Robotics, 2nd Edition*, p.74, Addison–Wesley, (1989).

3. Stroustrup, Bjarne, *The C++ Programming Language, 2nd Edition*, Addison–Wesley, (1991)